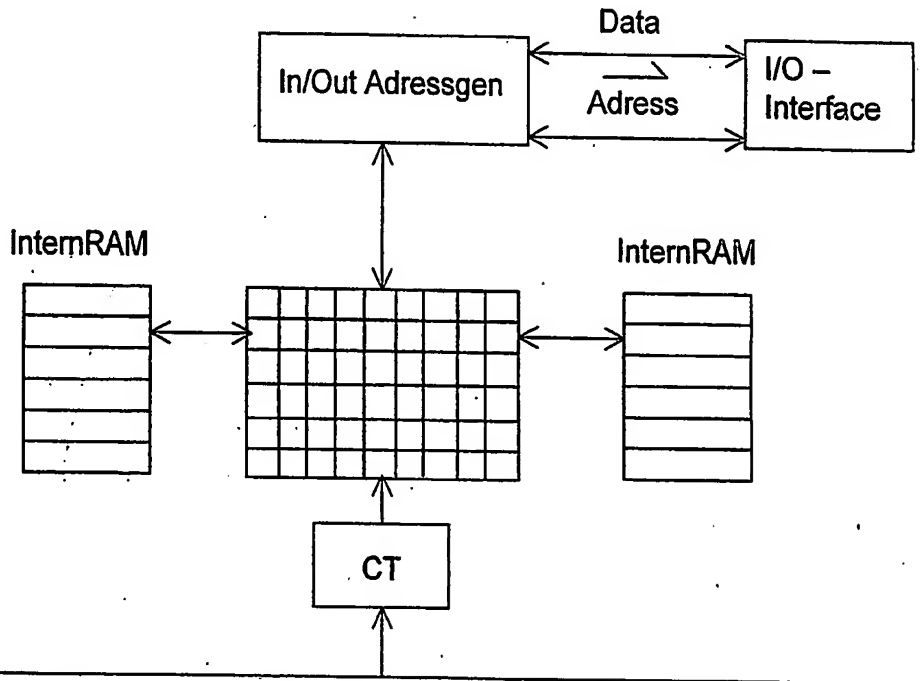


(Fig. 1b I, 1b II, 1b III) ⟷	config 1	Data@ ext.Adr.	→	Data@ IRAM-Adr.
(Fig. 1c) ⟷	config 2	Data@ IRAM-Adr.	→ „Ping“ →	Data´@ IRAM-Adr.2
(Fig. 1d) ⟷	config 3	Data´@ IRAM-Adr.2	→ „Pong“ →	Data^{n} @ IRAM-Adr.3
(Fig. 1f1, 1f2) ⟷	config n	Data^{n}´ @ IRAM-Adr.n	→	Result@ ext.Adr.2

Fig. 1a

10/551891

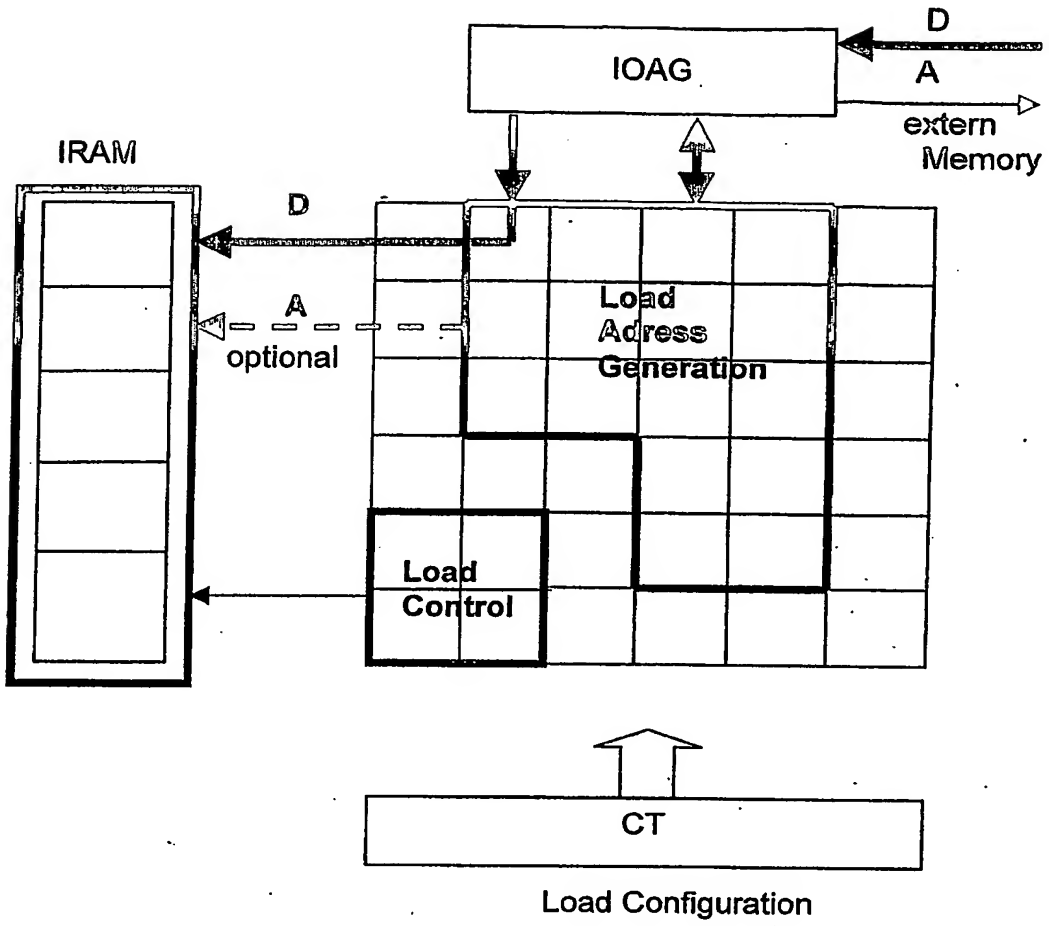

IOAG
D
A
extern
Memory
IRAM
D
A
optional
Load
Adress
Generation
Load
Control
CT
Load Configuration

Fig. 1b I

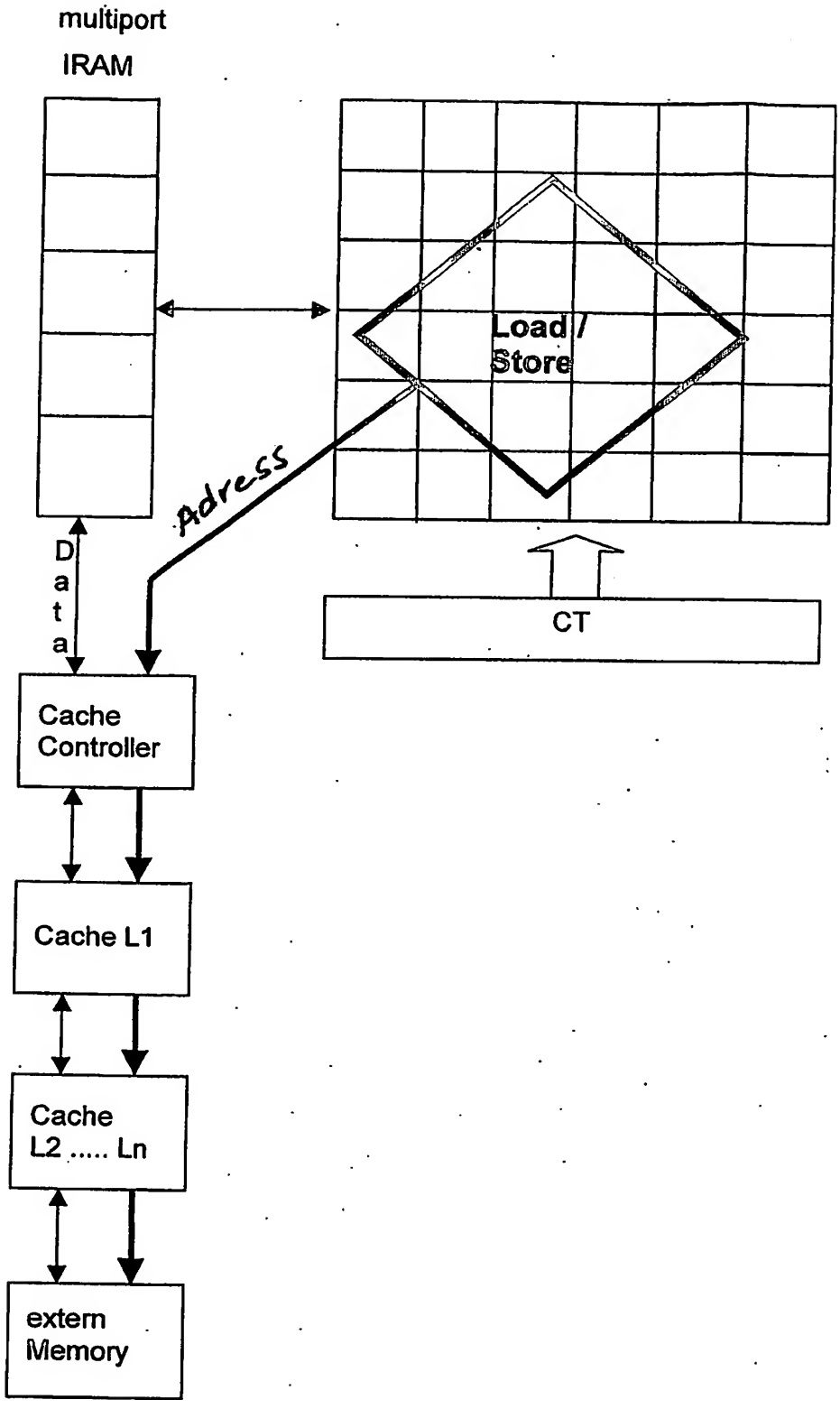

multiport
IRAM
Load /
Store
Adress
D
a
t
a
CT
Cache
Controller
Cache L1
Cache
L2 ..... Ln
extern
Memory

Fig. 1b II

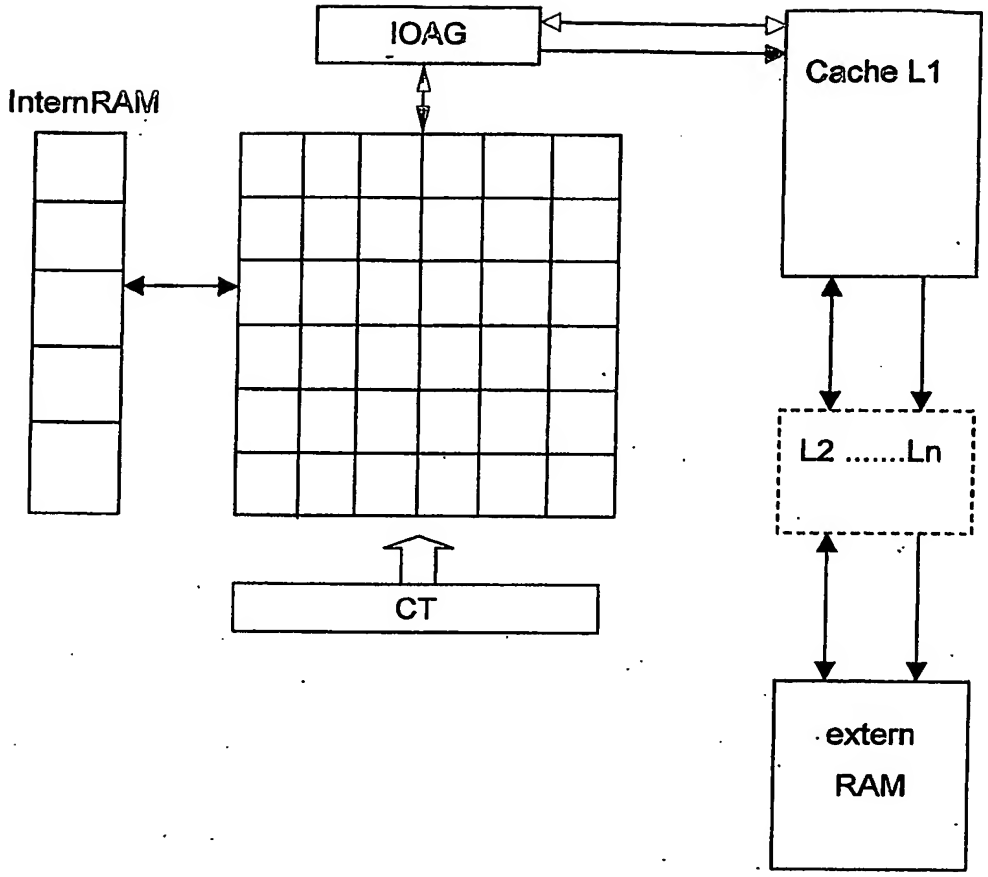
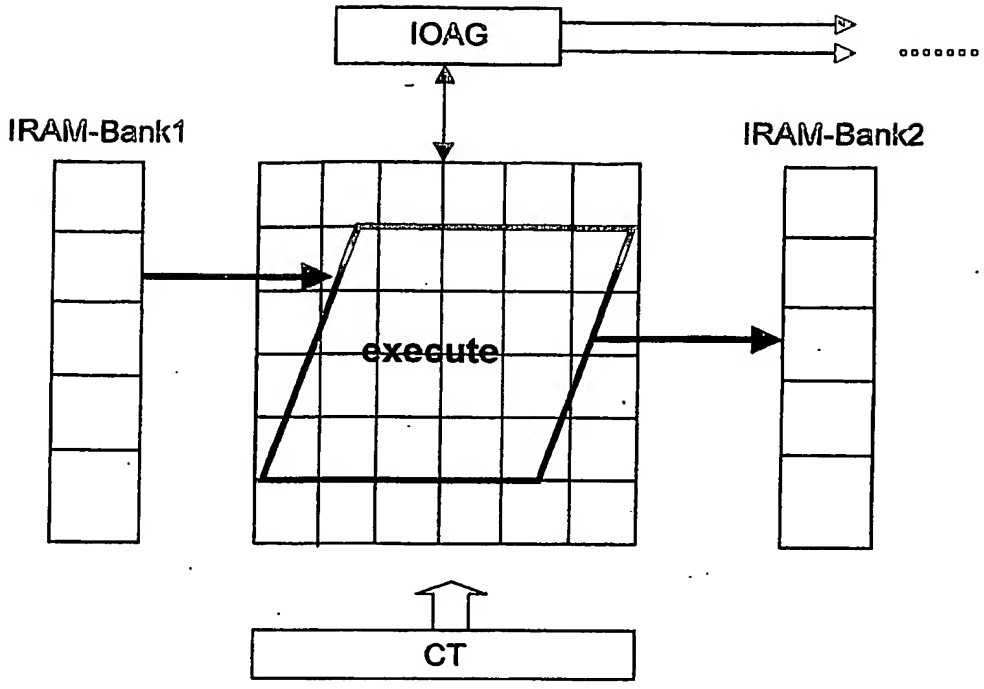
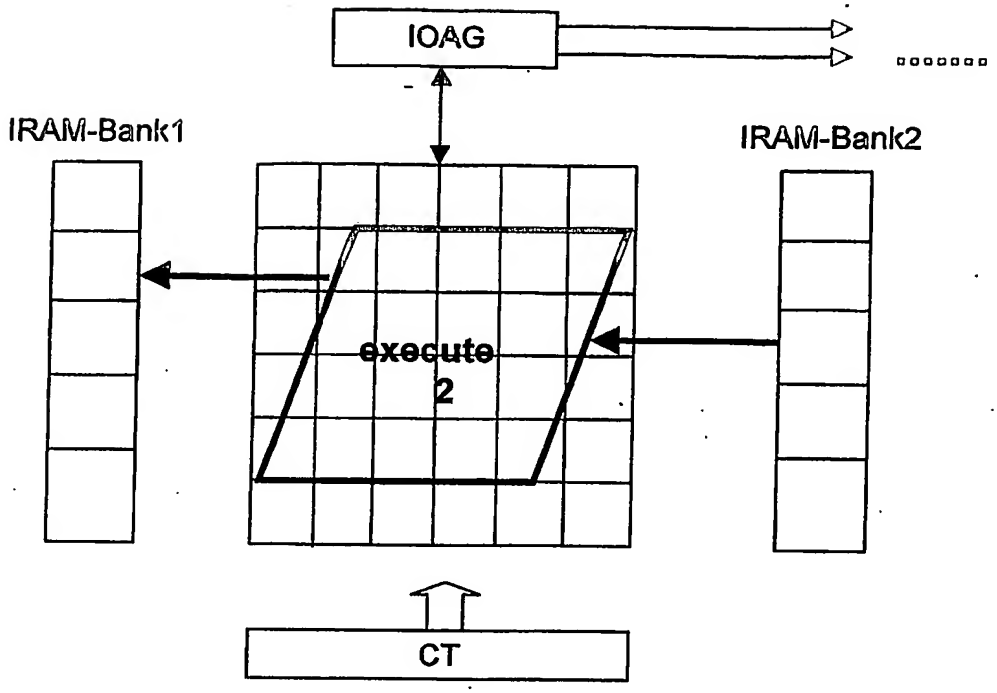


Fig. 1b III



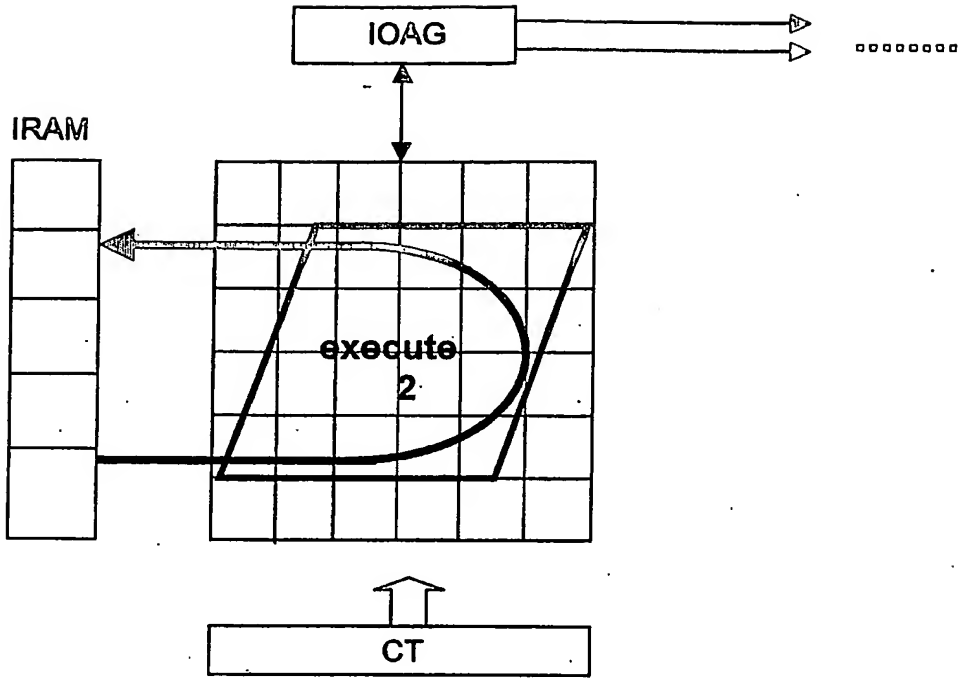
Ping

Fig. 1c



Pong

Fig. 1d



Ping / Pong

Fig. 1e1

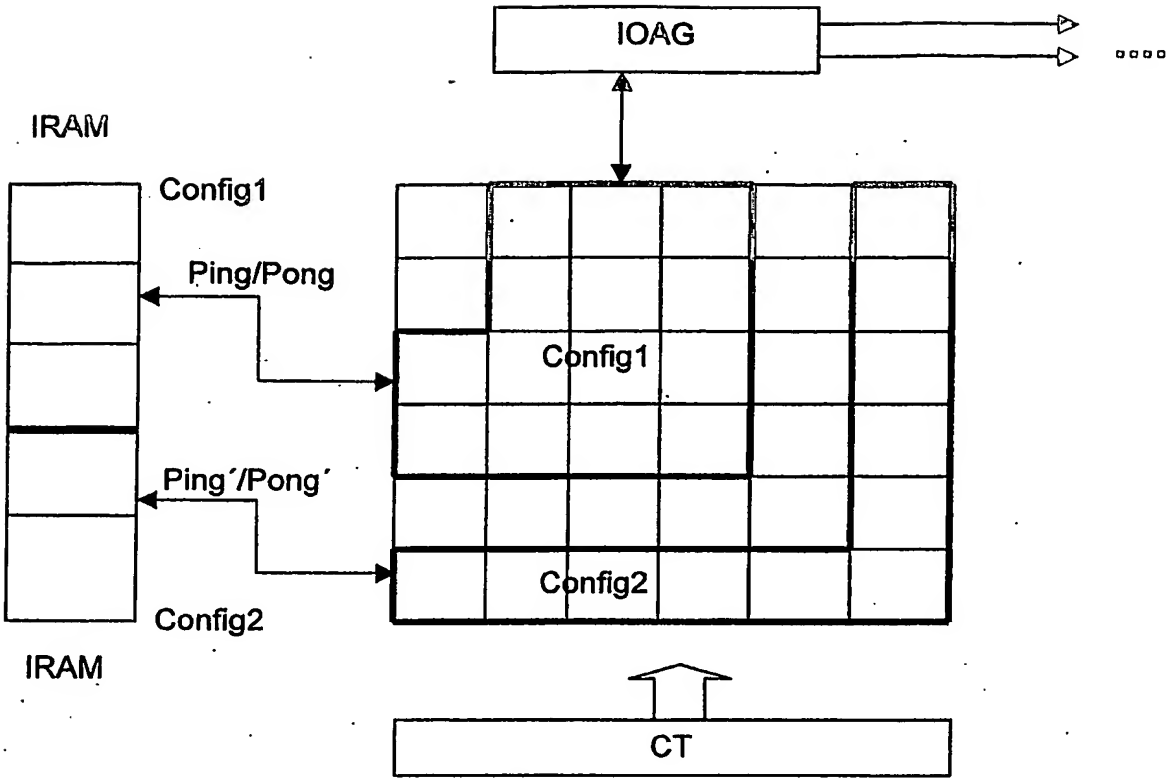

IOAG
IRAM
Config1
Ping/Pong
Config1
Ping´/Pong´
Config2
Config2
IRAM
CT

Fig. 1e2

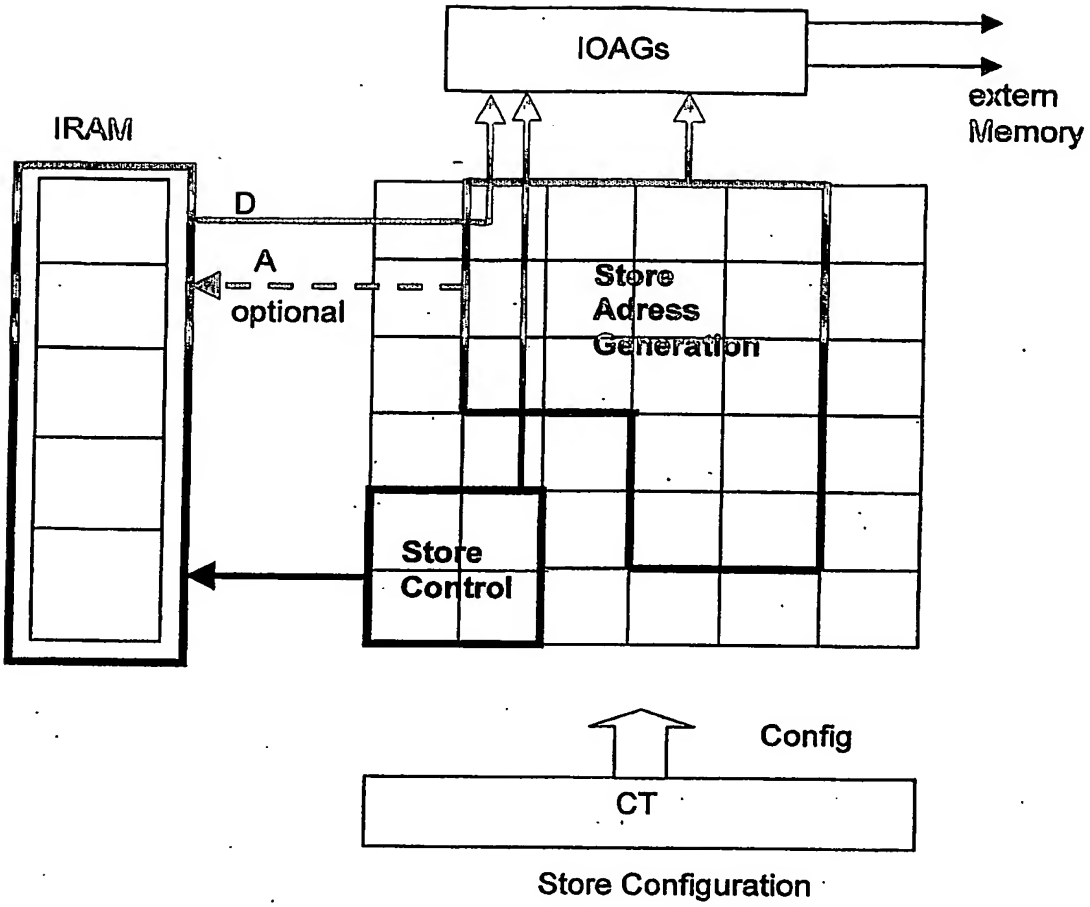

IOAGs
extern
Memory
IRAM
D
A
optional
Store
Adress
Generation
Store
Control
Config
CT
Store Configuration

Fig. 1f1

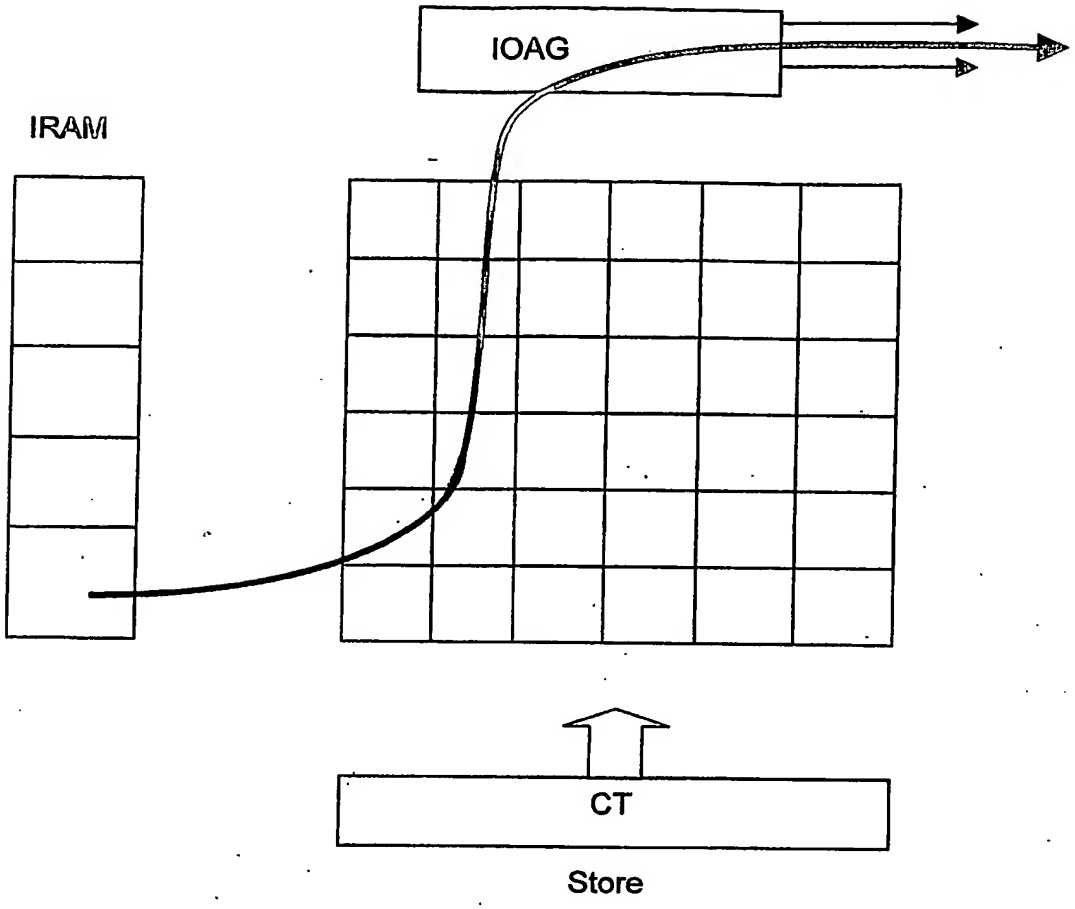


Fig. 1f2

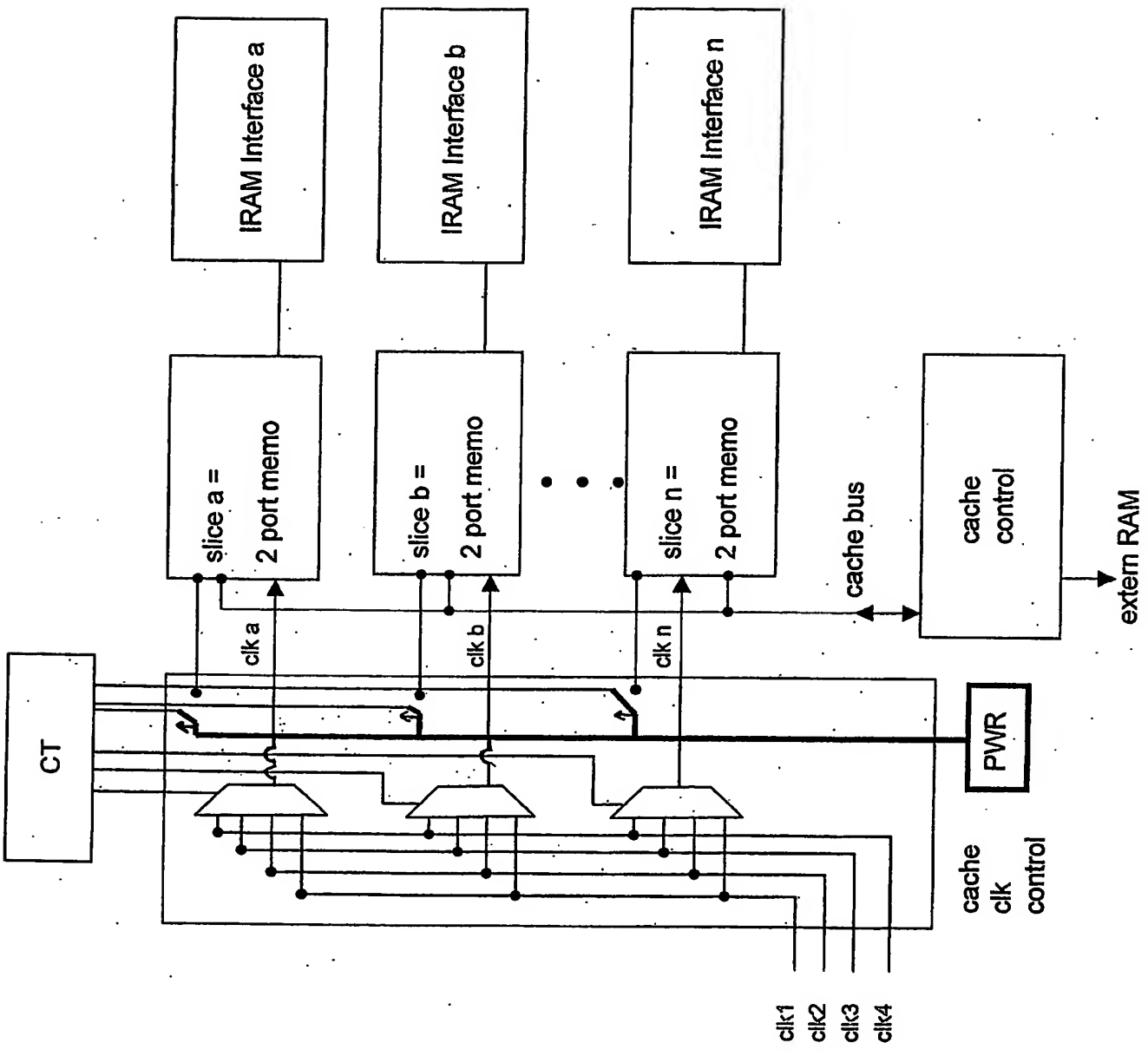

IRAM Interface a
IRAM Interface b
IRAM Interface n
slice a = 2 port memo
slice b = 2 port memo
slice n = 2 port memo
cache bus
cache control
extern RAM
CT
clk a
clk b
clk n
PWR
cache clk control
clk1
clk2
clk3
clk4

Fig. 2

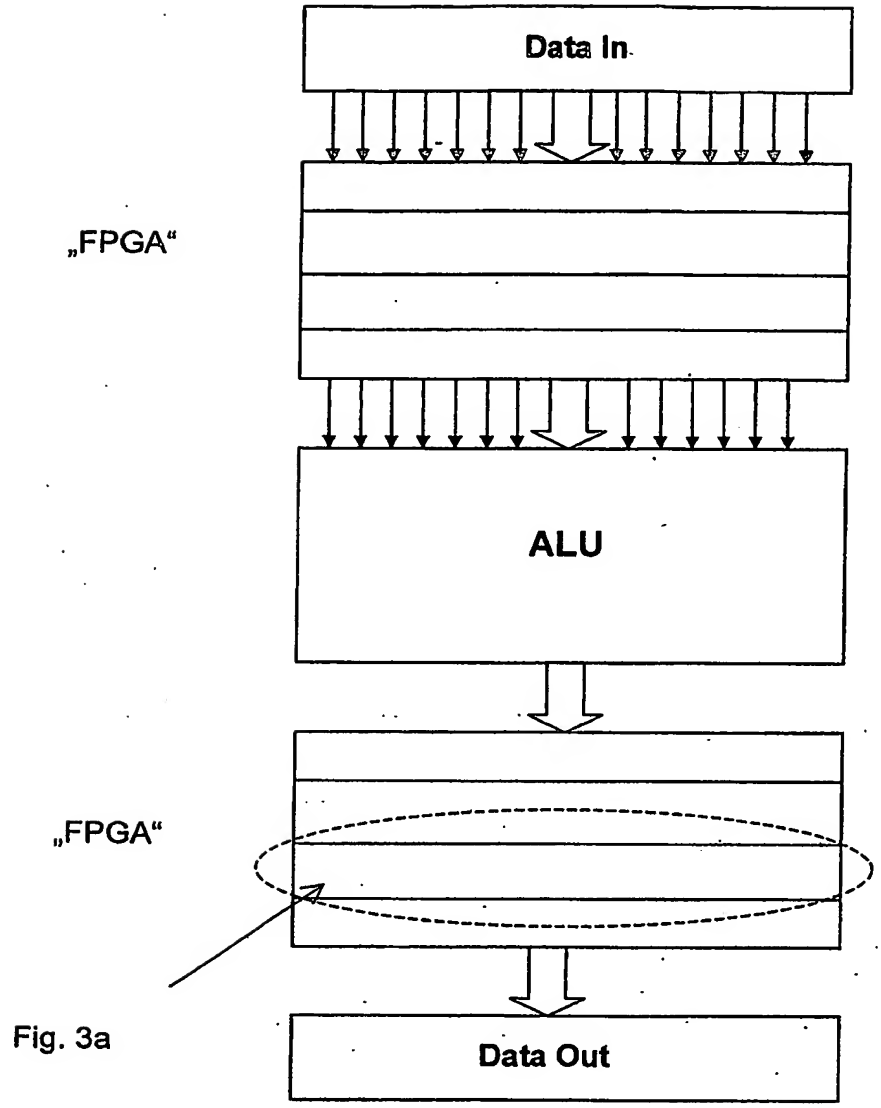


Fig. 3a

Fig. 3 I

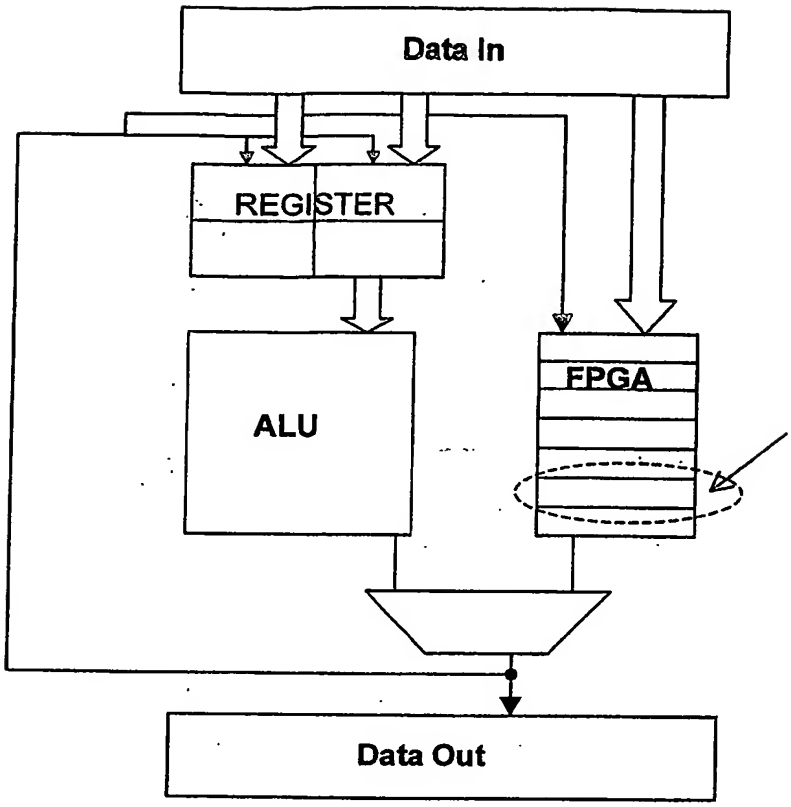


Fig. 3a

Fig. 3 II

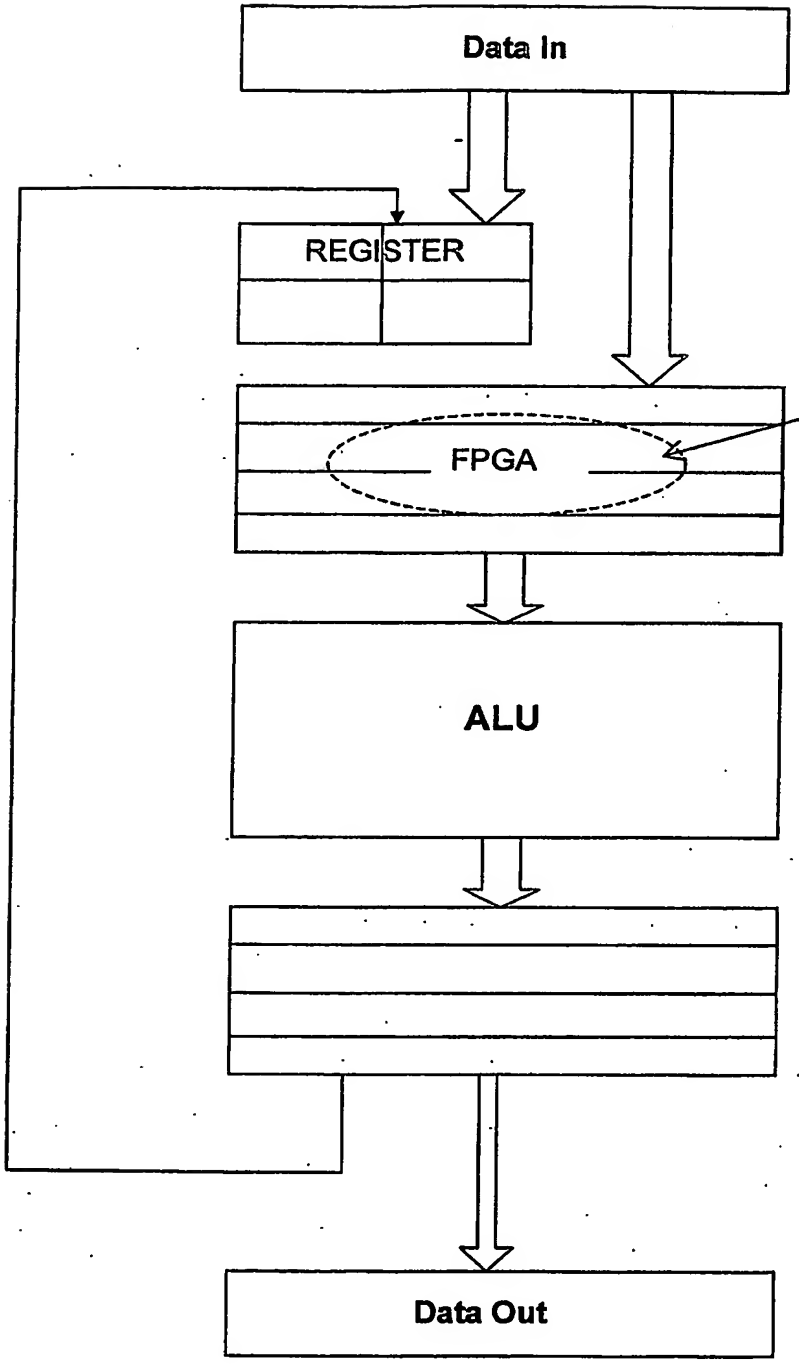


Fig. 3a

Fig. 3 III

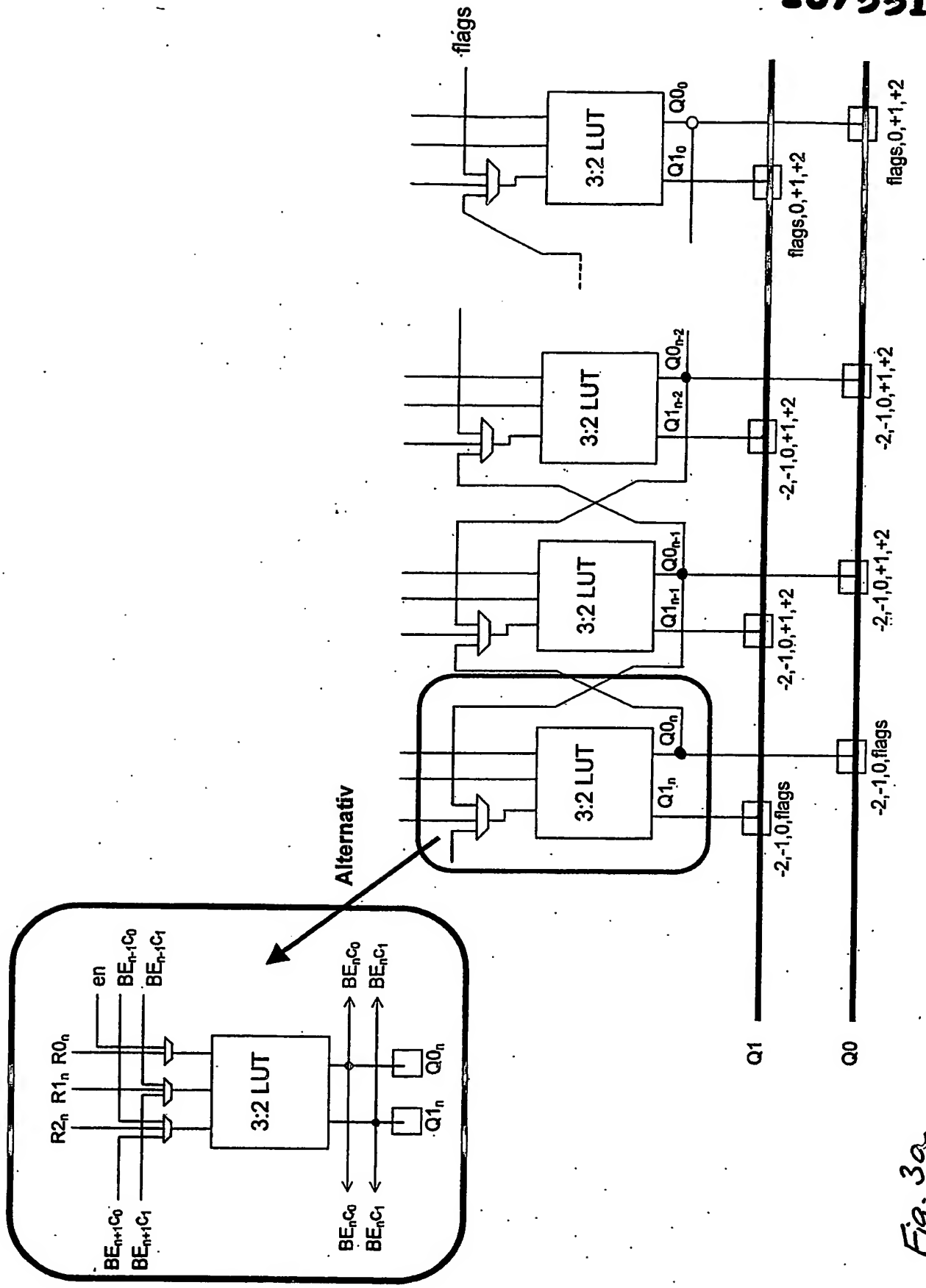
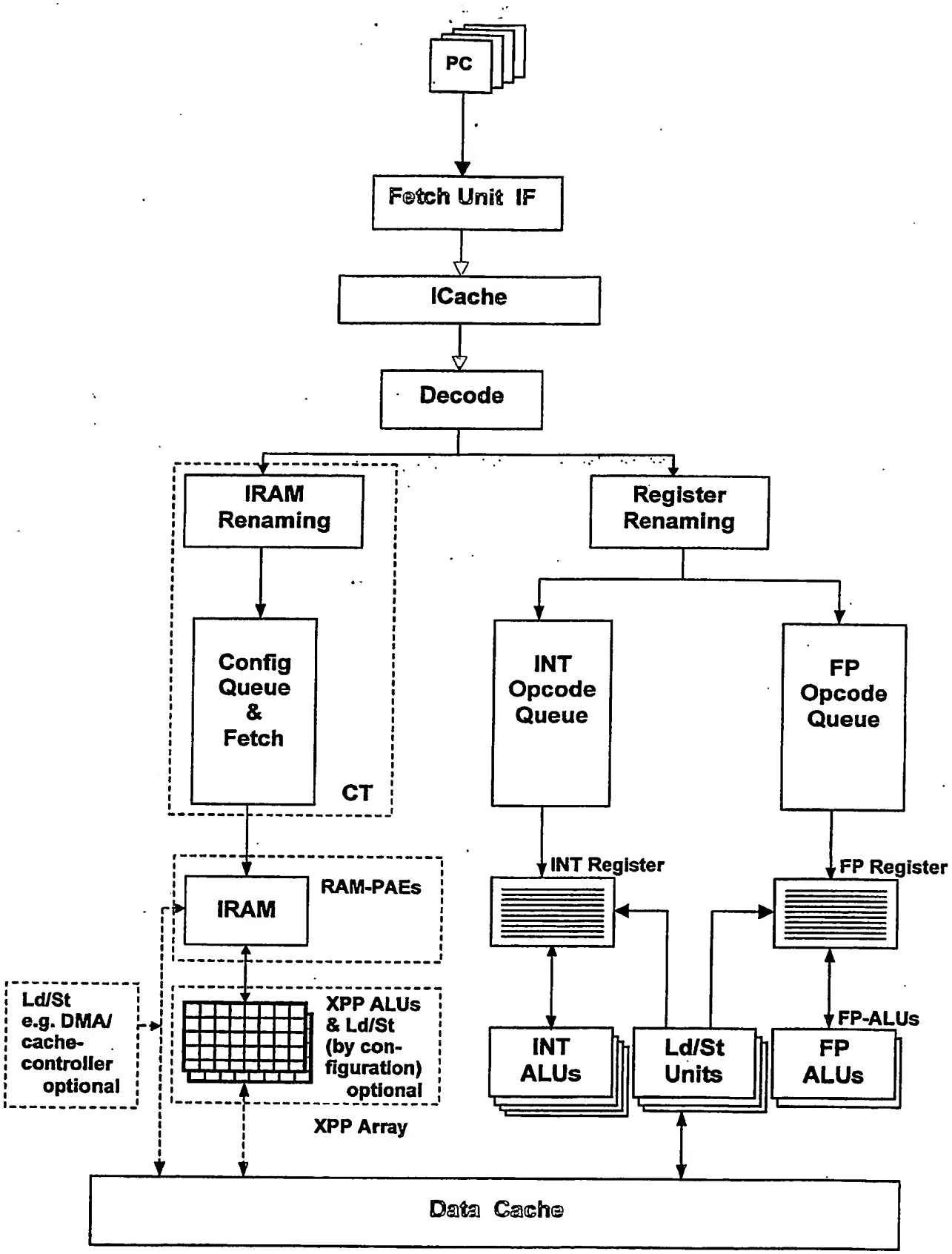

flags
3:2 LUT
Q0₀
Q1₀
flags,0,+1,+2
3:2 LUT
Q0n-2
Q1n-2
-2,-1,0,+1,+2
-2,-1,0,+1,+2
3:2 LUT
Q0n-1
Q1n-1
-2,-1,0,+1,+2
-2,-1,0,+1,+2
3:2 LUT
Q0n
Q1n
-2,-1,0,flags
-2,-1,0,flags
Q1
Q0
Alternativ
en
BEn-1c0
BEn-1c1
R2n R1n R0n
3:2 LUT
BEnc0
BEnc1
Q0n
Q1n
BEn+1c0
BEn+1c1
BEnc0
BEnc1

Fig. 3a



Möglicher Aufbau eines SMT-Prozessors mit XPP Thread Resource

Fig. 4a

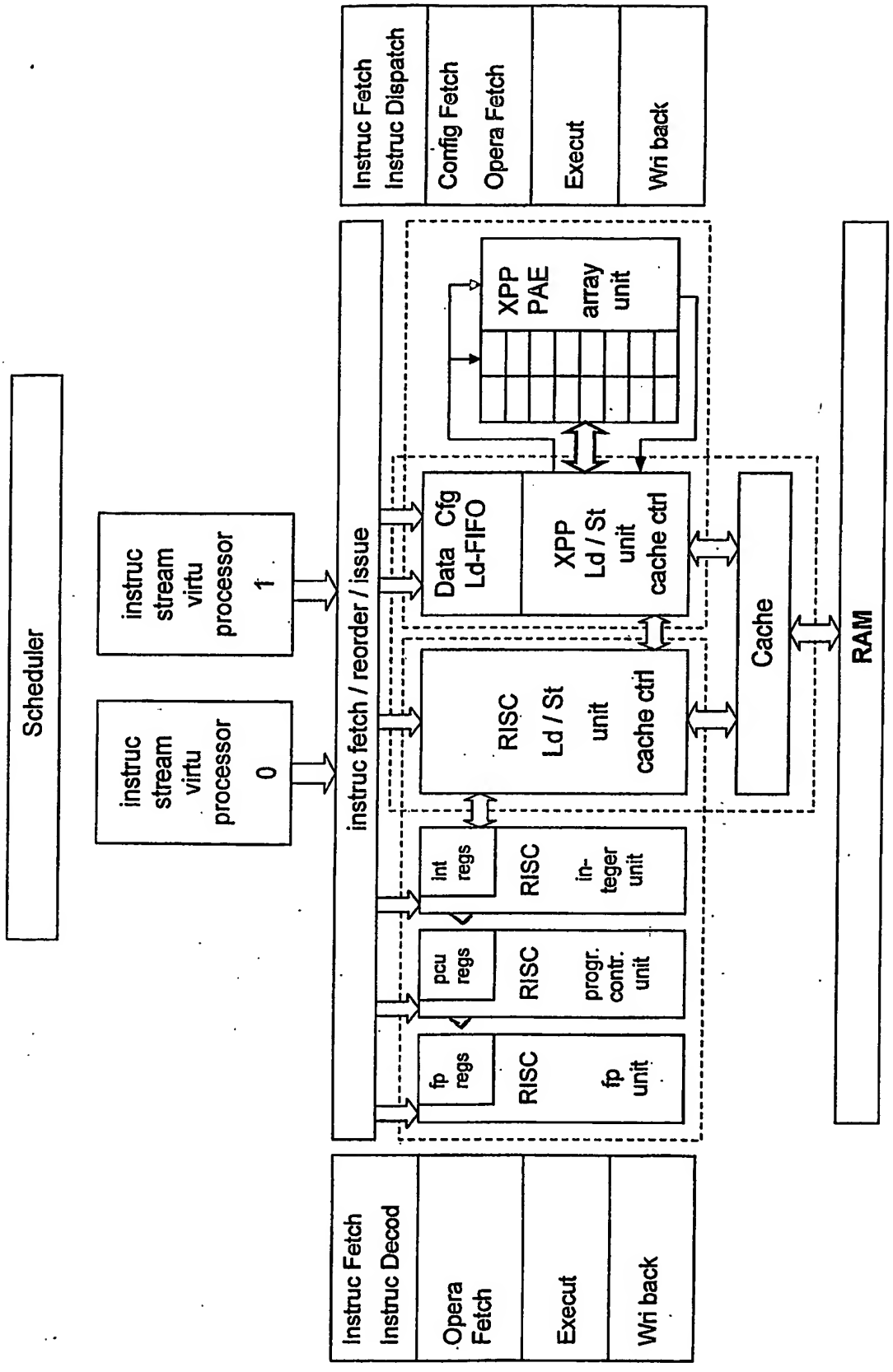

Scheduler
instruc stream virtu processor 0
instruc stream virtu processor 1
instruc fetch / reorder / issue
Instruc Fetch
Instruc Decod
Opera Fetch
Execut
Wri back
Instruc Fetch
Instruc Dispatch
Config Fetch
Opera Fetch
Execut
Wri back
fp regs
RISC
fp unit
pcu regs
RISC
progr. contr. unit
int regs
RISC
in-teger unit
RISC
Ld / St unit
cache ctrl
Data Cfg Ld-FIFO
XPP Ld / St unit cache ctrl
XPP PAE array unit
Cache
RAM

Fig 6

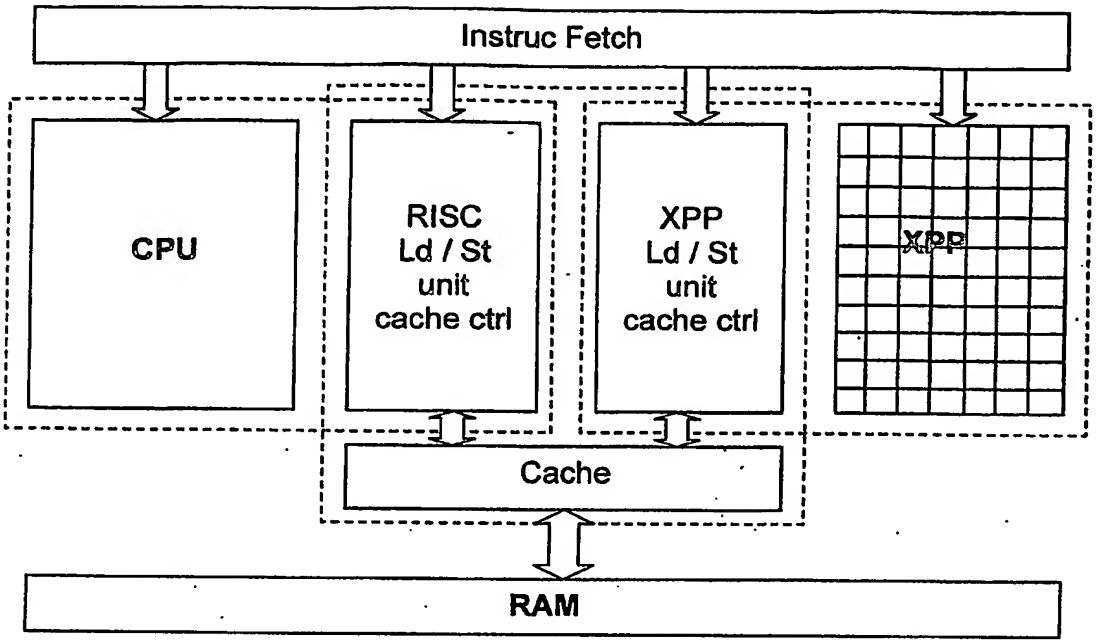

Instruc Fetch
CPU
RISC
Ld / St
unit
cache ctrl
XPP
Ld / St
unit
cache ctrl
XPP
Cache
RAM

Fig. 4C

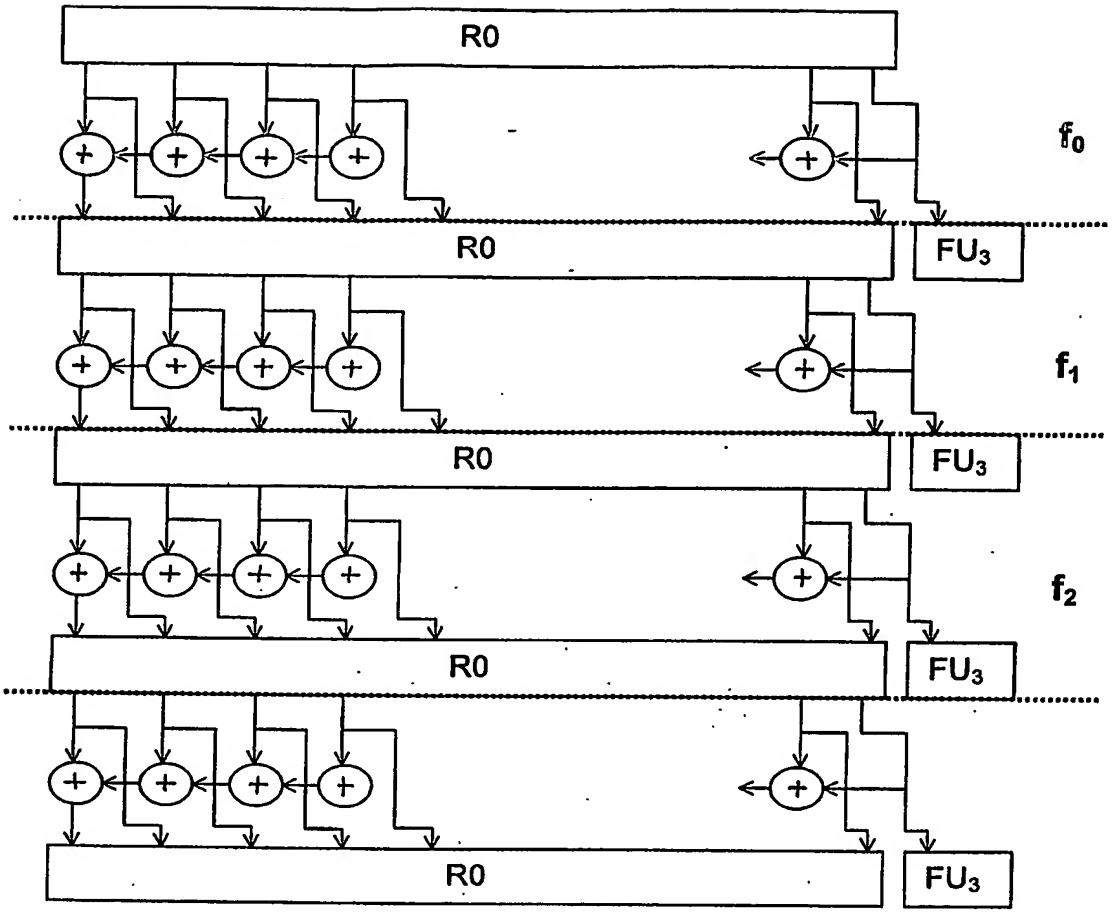
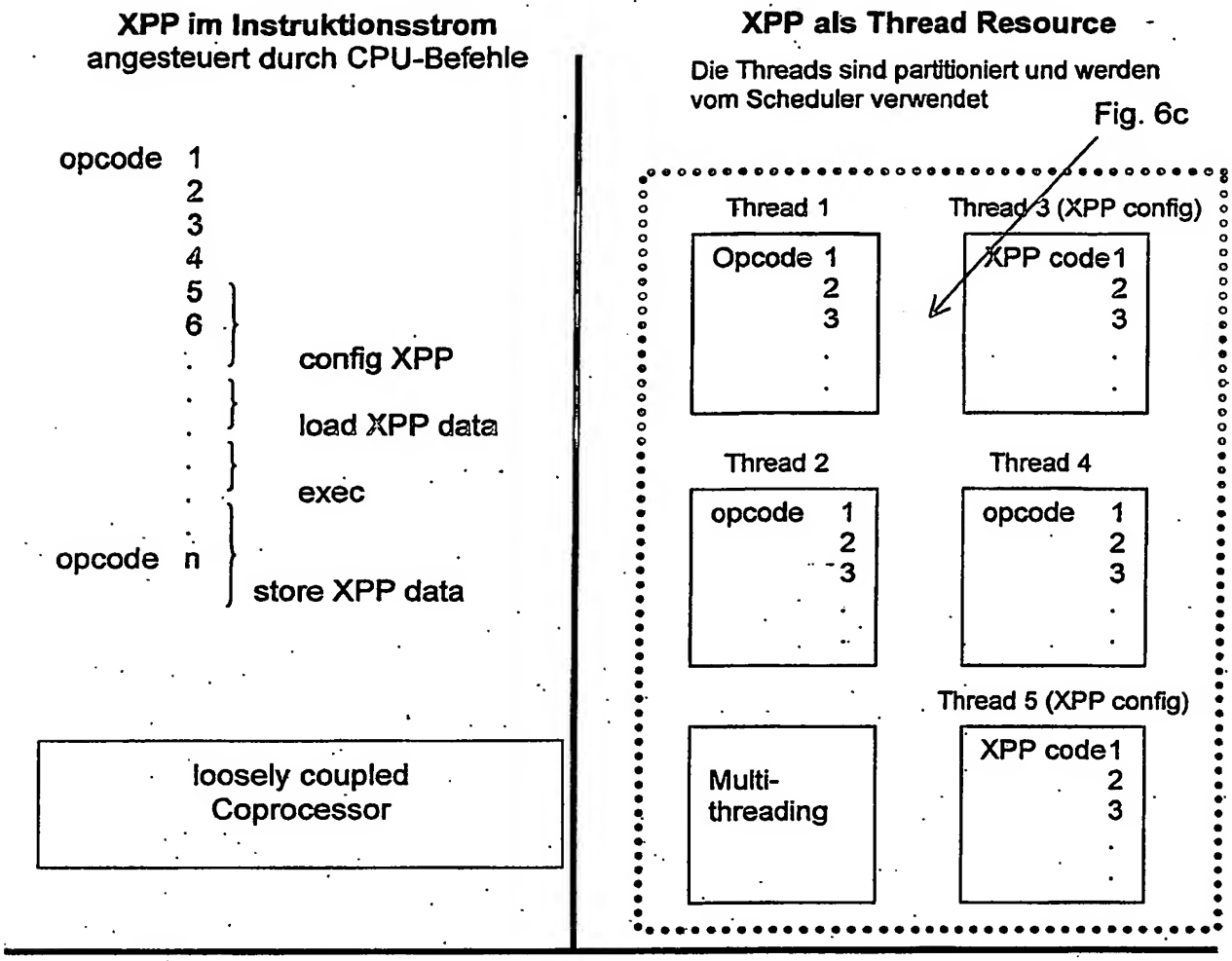


Fig. 5

	Scheduler	MEM-Interface	CT Operation	XPP Operation
1	Task 1	NOP	NOP	NOP
2	NOTask	Ld Task 1 Dat	Config Task 1	NOP
3	NOTask	NOP	NOP	Exec Task 1
4	NOTask	Stor Task 1 Dat	NOP	NOP
5	Task 2	Ld Task 2 Dat	Config Task 2	NOP
6	Task 3	Pre-Ld Task 3 Dat	Preconfig Task 3	Exec Task 2
7	Task 4	Pre-Ld Task 4 Dat Stor Task 2 Dat	Preconfig Task 4	Exec Task 3
8	NOTask	Stor Task 3 Dat	NOP	Exec Task 4

Fig. 6a

10/551891


XPP im Instruktionsstrom
angesteuert durch CPU-Befehle
opcode 1
2
3
4
5
6
config XPP
load XPP data
exec
opcode n
store XPP data
loosely coupled
Coprocessor
XPP als Thread Resource
Die Threads sind partitioniert und werden
vom Scheduler verwendet
Fig. 6c
Thread 1
Opcode 1
2
3
Thread 3 (XPP config)
XPP code1
2
3
Thread 2
opcode 1
2
3
Thread 4
opcode 1
2
3
Thread 5 (XPP config)
Multi-
threading
XPP code1
2
3

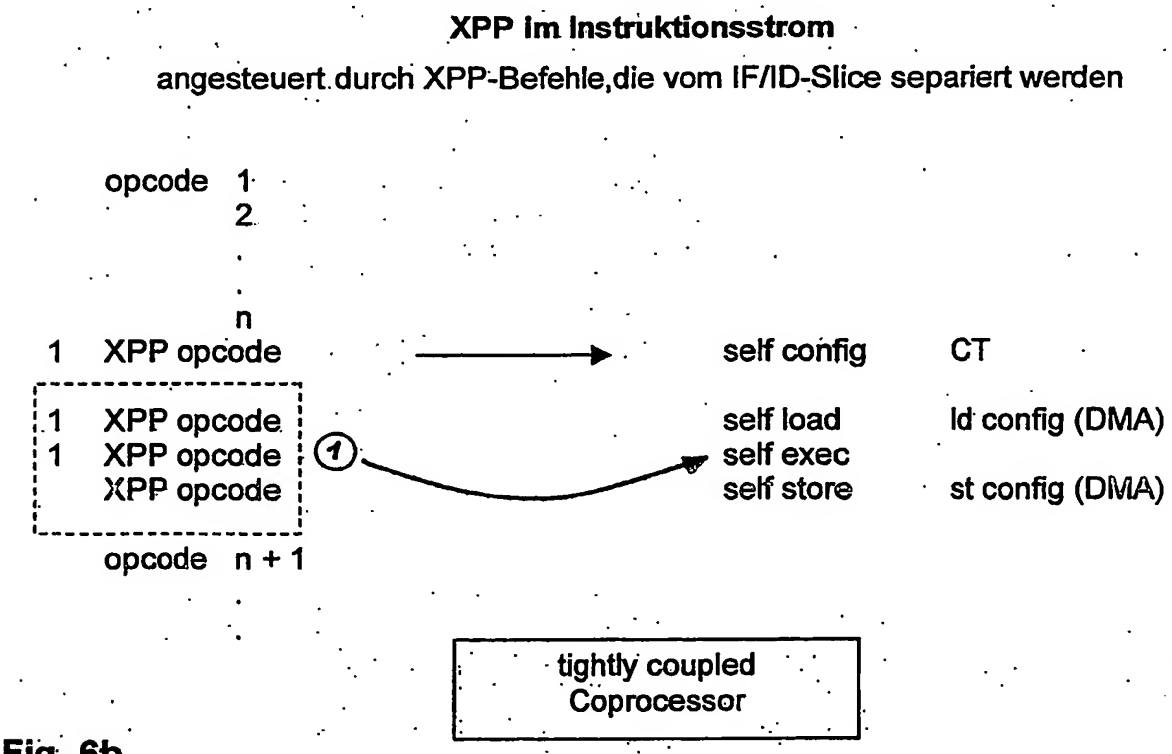

XPP im Instruktionsstrom
angesteuert durch XPP-Befehle,die vom IF/ID-Slice separiert werden
opcode 1
2
n
1 XPP opcode
1 XPP opcode
1 XPP opcode
XPP opcode
opcode n + 1
self config
CT
self load
ld config (DMA)
self exec
self store
st config (DMA)
tightly coupled
Coprocessor

Fig. 6b